PRO*forma*: an agent technology for decision support and plan enactment

John Fox, Nicky Johns, Ali Rahmanzadeh

Imperial Cancer Research Fund
Lincoln's Inn Fields
London WC2A 3PX UK
tel +44 171 269 3624
email {jf,nj,ahr,rt}@acl.icnet.uk

Abstract: PRO*forma* is a language and a set of software tools for defining medical decisions and procedures in a form which can be executed by a computer. The language is a first-order logic language which incorporates support for representing and reasoning about decisions and plans, based on a generalised agent architecture. The PRO*forma* toolset provides a graphical authoring environment for describing medical tasks, and an "enactment engine" which assists clinical staff in carrying out the processes of patient care. PRO*forma* was developed with medical applications in mind but the agent model and the toolset are generic and can be applied in other domains.

Keywords: Protocol-based care, decision support systems, safety, knowledge representation, knowledge based systems.

Introduction

It is widely thought that the practical potential of clinical decision support systems will only be achieved when standard languages for representing medical knowledge and clinical procedures are available. Various attempts have been made to do this, such as the "Arden syntax", a Pascal-like language, which was proposed as a standard format in which medical knowledge can be described as *Medical Logic Modules,* which can be embedded and reused in applications. Standardisation efforts like the Arden Syntax are important, but proposals to date have been criticised. For example, as Musen et al note [1], "this new standard has significant limitations: the language currently supports only atomic data types, lacks a defined semantics for making temporal comparisons or for performing data abstraction, and provides no principled way to represent clinical guidelines that are more complex than individual situation-action rules" [1].

PRO*forma* is being developed to address these limitations and to satisfy a number of additional requirements including clinical generality, medical expressiveness and operational flexibility.

An example application

A decision support system concerning the management of acute asthma in adults illustrates one kind of application that PRO*forma* is intended to support (figure 1). The workflow display shown in figure 1 can be used *passively*, to summarise and remind clinical staff of the procedures that need to be carried out when managing a patient with an acute asthma attack, or *actively*, to assist in the correct and timely enactment of specific clinical tasks. In the latter mode the system may prompt users for clinical actions which need to be carried out (taking into account temporal and other constraints), remind them of information that needs to be recorded, and advise on any decisions that need to be taken.

In the example the first task required by the guideline is a patient assessment decision (shown as a circle). The two panels on the right are an *aide memoire* that lists the

topics that are relevant to the assessment, and a data entry screen for the user to enter various patient data. Depending upon the assessment (mild, moderate, severe or life threatening) enactment of the procedure will proceed along the appropriate branch emerging from the assessment decision.

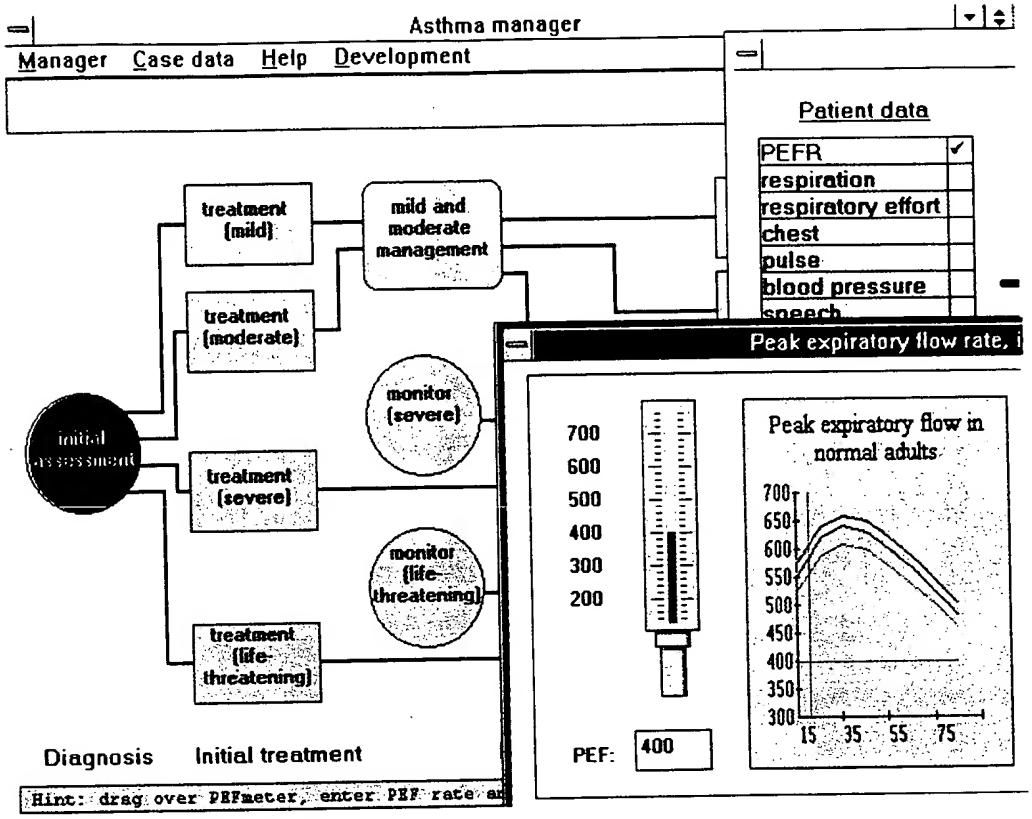

Figure 1: A user display from a computerised guideline/protocol management system, in this case for the management of acute asthma in adults in the accident and emergency department.

Decision support functions, which are used for the initial classification of the severity of the patient's asthma and the implementation of watchdogs, use a generalised decision procedure that can be used to support most types of clinical decision, including diagnostic, therapeutic and drug prescribing decisions. A range of different kinds of patient–specific advice can be provided by these functions, including suggesting when a decision is needed; what the decision options are; which information is relevant to making the choice and so on. A particularly crucial piece of advice is whether the decision can (and should) be taken and, if so, which is the best alternative of the available options given what is currently known.

PRO*forma* is a generic technology for building applications of this kind. It consists of the PRO*forma language,* a formal specification language in the sense used in software engineering, and a knowledge representation language as this term is used in AI. The technology also includes a number of software tools, for designing and "enacting" PRO*forma* applications.

Foundations of the PRO*forma* language: the domino model

The PRO*forma* language is based on a generalised model of the clinical care process: the "domino model". Figure 2 shows the model, in which the nodes in the domino

represent *concepts* and the arrows represent *inference processes*. The PRO*forma* technology supports data–structures representing the former and automated inference mechanisms representing the latter.

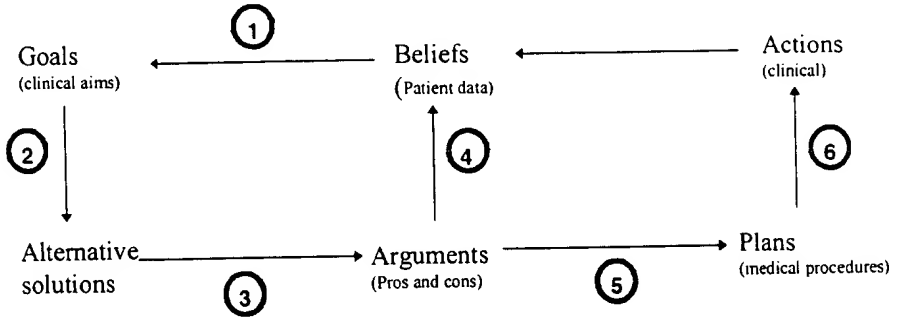

Figure 2: The domino agent model

The domino represents a generic structure for an "intelligent agent" which has beliefs, goals, plans and so forth, and various inference capabilities for problem solving and decision making.. Suppose data are added to a patient record, such as a statement that the patient has lost weight and that this appears to be abnormal (this is a "belief"). Normally a clinician would consider this to indicate a possible abnormality and therefore represent a clinical problem: to establish the cause of the abnormality (a goal). This inference process is represented by the arrow labelled (1) in the agent model.

From general medical knowledge the agent may now deduce that possible causes of weight loss include, say, cancer and peptic ulcer, and record these as *alternative solutions* for the diagnosis problem (2). Using general medical knowledge and information in the patient record (e.g. the patient is elderly) and knowledge about the kinds of information which are relevant in making diagnosis decisions (e.g. facts about physiology, statistical information), *arguments* can be constructed for and against each alternative diagnosis (3). General forms of diagnostic argument include "any condition which we know can *cause* the presenting problem is a possible diagnosis" or "when looking for arguments in favour of a disease consider any symptoms or signs in the patient record which are known to be frequently or rarely associated with each hypothetical disease".

At some point enough information may be available about the patient for the agent to recommend a decision to *commit* to one diagnosis candidate or another (4). Suppose it is decided that the patient is suffering from cancer. This conclusion is added to the patient record as a new belief.

This leads to an inference that the agent has another problem (arrow 1 again); how should the cancer be treated? As before, a set of candidate solutions is inferred from medical knowledge (2), though this time the candidates are not diagnoses but *medical procedures*, such as chemotherapy, radiotherapy or surgery. Once again, the pros and cons of these alternatives are argued (3) and, in due course, the agent proposes another decision, in this case to commit to a particular therapeutic procedure (5).

Execution of clinical procedures often involves a number of steps. For example, when a cancer patient has chemotherapy it is normal to obtain various baseline measurements on the patient (such as the patient's white blood count), and to administer several cycles of chemotherapy, usually some days or weeks apart. During each cycle the effects of treatment are monitored at intervals and this will be

continued for some time following therapy. Typically these *tasks* and the specific actions involved (like taking blood samples and giving medication) will need to be scheduled in a particular order and/or at particular times. The scheduling process (6) must be done using general knowledge of temporal and logical constraints, and situation–specific factors like the patient's well–being and availability of resources. Some tasks will be obligatory, some optional and some based on decisions if particular events occur or new problems arise.

The PRO*forma* technology is based on the domino agent model. It offers the necessary formalisms and software tools for implementing applications which can carry out the kinds of reasoning described in the example, and creating and maintaining the required data–structures.

Designing a clinical procedure with PRO*forma*

Three kinds of knowledge are required to build a PRO*forma* application: knowledge of the specific situation (about the patient, including personal details, problems, symptoms, current medications and so forth), ontological knowledge (of diseases, symptoms, tests, drugs), and knowledge of procedures (what should be done when). PRO*forma* is primarily focused on the last type of knowledge.

Creating a PRO*forma* application is a two–step process. The first step organises the application as a set of clinical tasks together with their logical and temporal interrelationships using a graphical design environment. In the second step the task network is converted into a database, using a knowledge editing tool, and populated with the detailed procedural and medical knowledge base required to execute the guideline.

Step 1: graphical representation of the high level structure of a guideline

The top level structure in PRO*forma* is the *task*. PRO*forma* applications are designed to provide support for a comprehensive range of clinical tasks, such as decision making, scheduling of actions over time, reminders for patient data collection, recording clinical events and so forth. PRO*forma* supports four basic classes of task as shown in (figure 3):

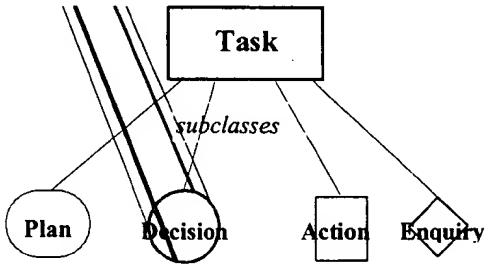

Figure 3: The PRO*forma task ontology*

- A *plan* is a sequence of sub–tasks, or components, which need to be carried out to achieve a clinical objective, such as a therapeutic objective. Plan components are usually ordered, to reflect temporal, logical, resource or other constraints.
- A *decision* task, such as a diagnosis or therapy decision, is modelled as a set of decision options, rules for arguing for and against the options, and "commitment rules" which determine when the decision should be made.
- An *action* task is a procedure which is "external" to the computer system, such as the task of administering an injection.

- An *enquiry* is a task whose objective is to obtain an item of information which is needed in order to complete a procedure or take a decision. The specification of an enquiry includes a description of the information required (e.g. a lab. result) and a method for getting it (e.g. by a query on a local patient record, or a remote laboratory database).

Figure 4 shows a view of the visual programming environment which has been developed for designing and implementing PRO*forma* applications.

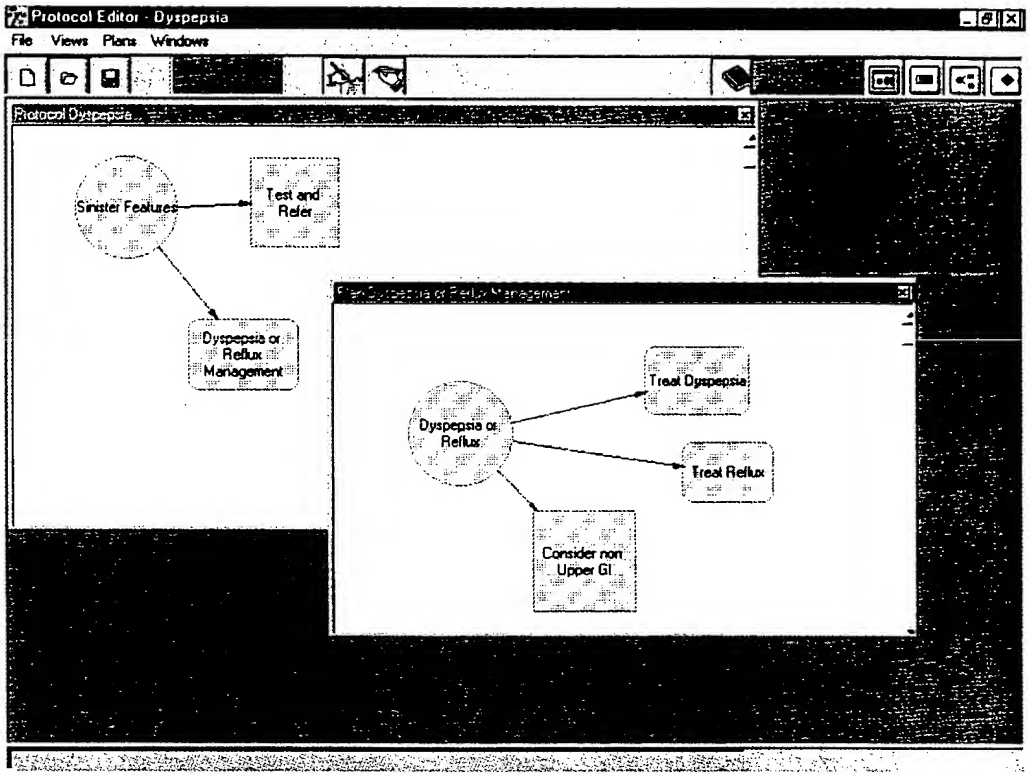

Figure 4: A view of the PROforma editor which provides graphical tools for laying out protocols in terms of four basic types of task; decisions (circles); plans (round rectangles); actions (square rectangles) and enquiries (diamonds). The arrows represent scheduling constraints. The rear window shows the component tasks in guideline for the management of dyspepsia in primary care, and the front shows the task decomposition for the task "dyspepsia and reflux management".

The graphical notation provides an intermediate representation between an informal description of the clinical process and a machine executable knowledge base. The notation provides a succinct and natural summary which can be understood by medical specialists, and also helps to guide the detailed specification of the tasks by the application designer.

Step 2: Populating task templates

In step 2 of the PRO*forma* method, the detailed clinical knowledge and other information required to populate the task network is added. All clinical tasks are different, but can be modelled in terms of classes which have a common, generic structure. All clinical procedures, for example, have eligibility conditions (preconditions) and conditions under which the procedure should be abandoned (abort conditions). Decisions, actions and enquiries can similarly be modelled as generic tasks. PRO*forma* provides a standard attribute–value template, predefined for each class of task, to support the guideline definition process. Templates guide the

application designer in formalising the necessary medical knowledge required for each task required by the guideline or protocol.

Each of the four sub–classes of PRO*forma* task has a class–specific internal structure, but each sub–class also inherits a number of attributes from the general class "tasks". All tasks have a clinical objective (goal) for example. Each task type also has a group of specific attributes which distinguishes it from tasks of other types. Table 1 shows the distinguishing attributes of decision tasks, for example.

Attribute	Description
Candidates	The alternative decision options (hypotheses, actions).
Argument rules	Rules defining "pros and cons" of different candidates.
Commitment rules	Rules based on arguments for committing to a candidate i.e. taking a decision.
Sources	Any information sources required.
Mandatory information	Information essential for commitment of decision.

Table 1: Distinguishing attributes of decision tasks

From table 1 we see that all decisions (whether diagnostic decisions, therapy decisions, prescribing decisions, referral decisions, etc.) must have a set of options (candidates); they all have information sources that are relevant to choose between the options, rules of argument to establish preferences among the options, and commitment rules.

Plan enactment by the PRO*forma* Engine

The purpose of the PRO*forma* engine is to *enact* the tasks specified by a clinical guideline or protocol, that is to say it issues requests for action, prompts for information, suggestions for decisions, and generally assists a user to carry out the intended plan.

When the engine enacts a plan, each task in the protocol is "created" as needed as a run–time instance in memory with current time–stamps and other relevant information. The engine initiates tasks according to their scheduling constraints, preconditions etc., and sends appropriate messages to notify other agents that tasks have been activated, that decisions which need to be made, data acquired or actions carried out etc. Figure 5 shows the engine running a simple guideline for the management of indigestion.[1]

The user interface shown here is the default interface provided by the software, which is normally expected to be linked to a separate and appropriately tailored application–specific GUI, as in the asthma example above.

[1] This guideline, which is intended for use by general practitioners, was developed by Drs Colin Lyons and Peter Wilson of North End Medical Centre, London.

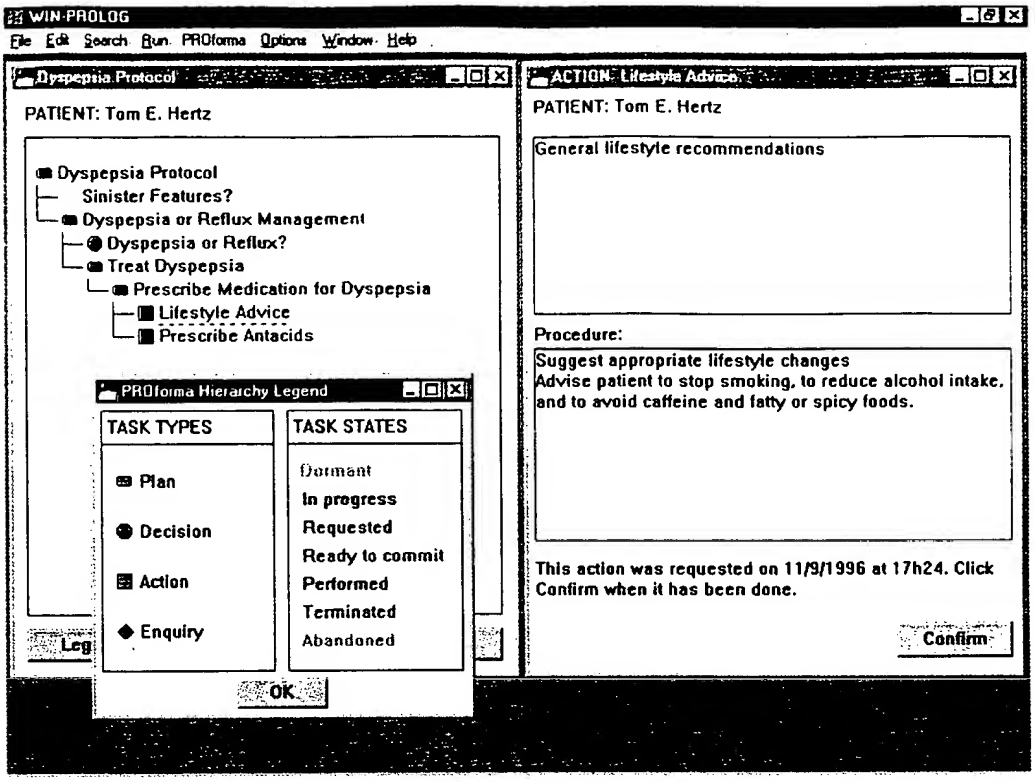

Figure 5: the default interface of the enactment engine, showing the hierarchical structure of a guideline for the management of dyspepsia (left), a legend indicating the types of task and colour coding for task states (inset) and a report for a selected task (right).

The engine functions reactively, i.e. it responds to messages from external sources (e.g. notifying it of commands from a user interface, updates to the patient record, the completion of time–outs etc.). Any arrival of new data will generally have repercussions throughout the current protocol (e.g. decisions which may now be taken, tasks which may now be started, etc.). The engine's response to the arrival of a message is to propagate all these effects forward until no more changes can be made.

Conclusion

PRO*forma* embodies a language and a development method for specifying clinical proc–edures such as therapeutic guidelines and protocols. It provides a notation for formally specifying a wide range of clinical tasks, and also the medical knowledge and patient data associated with clinical decision making. A fuller description of PRO*forma*, and its use in supporting guideline–based healthcare is available in [5]. Details of parts of the work underlying the development of PRO*forma* can be found in [2,3]. A detailed definition of the PRO*forma* language will be published in due course.

Acknowledgements

PRO*forma* is being developed within PROMPT *(Protocols for Medical Procedures and Therapies)*, a project supported by the EC 4th Framework Health Telematics programme (1996–8). The work is being undertaken by the ICRF in association with members of the ACTION cluster of projects. Thanks are due to Ian Herbert, Jean–Louis Renaud–Salis, Johan van der Lei and Paul Taylor for their contributions to the ideas reported in this paper.

References

1. Musen M A, Tu S W, Das A K and Shahar Y 'A component based architecture for automation of Protocol–Directed Therapy', in Barahona P, Stefanelli M and Wyatt J (eds) *Proc. AIME95*, Berlin: Springer, 1995.
2. Fox J et al 'LEMMA: methods and architectures for logic engineering in medicine' in J Noothoven and J P Christensen (eds) *Advances in Medical Informatics*, Amsterdam: IOS Press, 1992.
3. Das S K, Fox J, Elsdon D and Hammond P 'A generalised architecture for intelligent agents' *Journal of Experimental and Theoretical AI*, (in press).
4. Fox J, Johns N, Rahmanzadeh A and Thomson R 'The PRO*forma* decision support engine', PROMPT deliverable D5.1, ICRF Advanced Computation Laboratory, 1996.